Transport Layer Security (TLS) Extensions for the Trusted Platform Module (TPM)
draft-latze-tls-tpm-extns-01

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on May 31, 2010.

Copyright Notice

Copyright (c) 2009 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents in effect on the date of publication of this document (http://trustee.ietf.org/license-info). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.
Abstract

Trusted Platform Modules (TPMs) become more and more widespread in modern desktop and laptop computers and provide secure storage and cryptographic functions. As one nice feature of TPMs is that they can be identified uniquely, they provide a good base for device authentication in protocols like TLS. This document specifies a TLS extension that allows to use TPM certified keys with TLS in order to allow for a secure and comfortable device authentication in TLS.

Table of Contents

1. Introduction .................................................. 3
2. Terms and Abbreviations ..................................... 3
3. Certification Process .......................................... 3
4. TLS TPM Extension Type ....................................... 4
5. TLS TPM Supplemental Data Handshake Message ............... 5
6. TLS Handshake Using The TPM Extensions ..................... 6
7. IANA Considerations ........................................... 8
8. Security Considerations ....................................... 8
9. Acknowledgements ............................................. 8
10. Normative References ......................................... 9
Authors’ Addresses ................................................. 9
1. Introduction

This document aims at specifying a new TLS extension that allows to use TPM certified keys directly with TLS [RFC5246]. TPM is short for Trusted Platform Module and describes a trusted module that provides secure storage and some cryptographic function and has been specified in [TPMMainP1]. The TPM comes with the possibility to create so called Attestation Identity Keys (AIKs) that prove that a platform equipped with a TPM is a given platform. Although those AIKs cannot be used in protocols like TLS without further changes to the protocol, [TPMMainP3] introduces so called certified keys. Certified keys are RSA keys that are certified by other keys, for instance by an AIK. Keys that are certified by an AIK are non migratable which means they remain in the same TPM forever. In order to use those keys with TLS, one has to create a self-signed certificate including the SKAE extension [SKAE], which will be used during the TLS handshake. In order to be able to verify that the key is stored inside a given TPM, the AIK will be send in the supplemental data handshake message.

2. Terms and Abbreviations

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Furthermore, the document uses the following terms and abbreviations:

AIK - Attestation Identity Key

CA - Certificate Authority

Entity - One of the communication end points, be it either client or server.

PCA - Privacy CA

TLS - Transport Layer Security

TPM - Trusted Platform Module

3. Certification Process

This section describes the process of creating and requesting all the certificates necessary to be used with the TLS TPM extension specified in the next sections.
First of all an TPM equipped entity has to request its AIK as specified in [TPMMainP1]. Afterwards, a new non-migratable key has to be created and certified using the AIK. The details about the certification process can be found in [TPMMainP3]. Some details will be repeated here for convenience.

The certificate is done using either TPM_CertifyKey or TPM_CertifyKey2 (for details about when to use which function, have a look at [TPMMainP3]). Depending on the key properties, those functions result in either TCPA_CERTIFY_INFO or TCPA_CERTIFY_INFO2 structure, whereas the first is compatible to the TPM 1.1 standard [TCGMainSpec].

Now that the entity has an AIK and a certified key structure, a self-signed certificate around the certified key has to be created. As the TCPA_CERTIFY_INFO (or TCPA_CERTIFY_INFO2) structure is needed to verify the binding between AIK and the certified key, that self-signed certificate has to include the Subject Key Attestation Evidence (SKAE) extension defined in [SKAE]. The SKAE extension is an X.509 extension that has been defined to carry the certify info structure returned by TPM_CertifyKey (or TPM_CertifyKey2).

The self-signed certificate will be sent during the TLS handshake in the Certificate message whereas the AIK MUST be announced with the TLS TPM extension type and sent in the supplemental data handshake message.

4. TLS TPM Extension Type

The general TLS extension format has been defined in [RFC5246] and will be repeated here for convenience:

```c
struct {
    ExtensionType extension_type;
    opaque extension_data<0..2^16-1>;
}
```

The new extension types for TPM enabled entities are called client_aik and server_aik:

```c
enum {
    client_aik(TBD), server_aik(TBD), (65535)
} ExtensionType
```

This extensions MAY be used in full handshakes as well as in session resumption handshakes. Although the latter does not require a certificate exchange it might happen that the server refuses to accept a resumed session and runs a full handshake instead. In order to be able to do that without interruption, the extensions SHOULD be
The extension includes the certify info type the client is able to create and verify:

```c
enum {
    tpm_certify_info(0), tpm_certify_info2(1), (255)
} CertifyInfoType
```

The client includes client_aik in order to indicate that he wants to use a self-signed certified key during the handshake and send the AIK in the supplemental data handshake message. If the server receives client_aik, he MUST respond with same client_aik - possibly removing unsupported certify info types or omit the extension in case it is not supported by the server.

In case the client wants to authenticate the server also using TPM certified keys, he MUST include server_aik in its extended hello message. The server_aik contains all the certify info types the client is able to verify. If the server receives server_aik and accepts it, he MUST respond with the same server_aik - possibly removing certify info types he cannot create. Otherwise the server omits server_aik.

### 5. TLS TPM Supplemental Data Handshake Message

The TLS supplemental data handshake message as defined in [RFC4680] allows to send additional application data during the TLS handshake if it has been announced in a TLS extension.

This document defines a new supplemental data type:

```c
enum {
    aik_data(TBD), (65535)
}
```

with

```c
struct {
    SupplementalDataType supplemental_data_type;
    select(SupplementalDataType) {
        case aik_data: AikData;
    }
} SupplementalData
```

and

```c
opaque ASN.1Cert<2^24-1>;
```

```c
struct {
    ASN.1Cert certificate_list<0..2^24-1>;
} AikData;
```
AikData carries the entity’s AIK chain.

6. TLS Handshake Using The TPM Extensions

Figure 1 shows the full TLS handshake with a TPM equipped client:

```
Client                                 Server
ClientHello (w/ extensions)--------->  ServerHello (w/ extensions)
                                    Certificate
                                    ServerKeyExchange
                                    CertificateRequest*
                                    <-------- ServerHelloDone

SupplementalData
Certificate*
ClientKeyExchange
CertificateVerify*
ChangeCipherSpec
Finished                   --------->
ChangeCipherSpec
Finished
```

* indicates optional or situation dependant messages

Figure 1: Full TLS Handshake With a TPM Equipped Client

Figure 2 shows the full handshake with a TPM equipped server:
Finally, figure Figure 3 shows the TLS handshakes if both sides make use of certified keys:

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>ClientHello (w/ extensions)</td>
<td>ServerHello (w/ extensions)</td>
</tr>
<tr>
<td></td>
<td>SupplementalData</td>
</tr>
<tr>
<td></td>
<td>Certificate</td>
</tr>
<tr>
<td></td>
<td>ServerKeyExchange</td>
</tr>
<tr>
<td>Certificate*</td>
<td>CertificateRequest*</td>
</tr>
<tr>
<td></td>
<td>ServerHelloDone</td>
</tr>
<tr>
<td>SupplementalData</td>
<td></td>
</tr>
<tr>
<td>Certificate*</td>
<td></td>
</tr>
<tr>
<td>ClientKeyExchange</td>
<td></td>
</tr>
<tr>
<td>CertificateVerify*</td>
<td></td>
</tr>
<tr>
<td>ChangeCipherSpec</td>
<td></td>
</tr>
<tr>
<td>Finished</td>
<td>ChangeCipherSpec</td>
</tr>
<tr>
<td></td>
<td>Finished</td>
</tr>
</tbody>
</table>

* indicates optional or situation dependant messages

Figure 3: Full TLS Handshake With TPM Equipped Client and Server

The authentication of either client or server is done by verifying the self-signed certificate as well as by verifying the binding...
between the AIK and the certified key in order to ensure that the key used is really protected by a given TPM. In order to verify the binding, the SKAE extension of the self-signed certificate has to be evaluated using the AIK.

There is no need for additional TLS alerts since all the existing certificate related alerts cover possible problems during the entity verification.

7. IANA Considerations

This document makes the following IANA requests:

1. A new registry for certify info types needs to be maintained by IANA. The first two types include tpm_certify_info(0) and tpm_certify_info2(1). Certify info types with values in the inclusive range of 0 to 63 (decimal) are assigned using RFC 5226 [RFC5226] Standards Action, whereas values from the inclusive range of 64 to 223 (decimal) are using RFC 2434 Specification Required. Values in the inclusive range of 224 to 255 (decimal) are reserved for RFC 2434 Private Use.

2. The values client_aik(TBD) and server_aik(TBD) are assigned from TLS Extension Type Registry [RFC5246].

3. The value aik_data(TBD) is assigned from TLS Supplemental Data Type registry [RFC4680].

8. Security Considerations

If an entity certified several keys with the same AIK, somebody who has the AIK and all of the certified keys is able to track that identity. Therefore, the AIK might be seen as sensitive information forcing an implementation to use the double handshake technique. The first handshake requires one or both entities to accept the self-signed certificate since the binding can only be verified during the second protected handshake.

9. Acknowledgements

The basic idea to use the supplemental data handshake message to supply the AIK was supplied by Sam Hartmann.
10. Normative References


Authors’ Addresses

Carolin Latze
University of Fribourg
Boulevard de Perolles 90
Fribourg, FR 1700
Switzerland

Email: carolin.latze@unifr.ch